RPyR : Nouvelle Structure d'Indexation avec Classification par Projections Aléatoires

Thierry Urruty, Fatima Belkouch, Chaabane Djeraba, Dan A. Simovici

RPyR : Nouvelle Structure d'Indexation avec Classification par Projections Aléatoires

T. Urruty[*] - F. Belkouch[*] - C. Djeraba[*] - D. A. Simovici[**]

[*] *LIFL-UMR CNRS 8022, Laboratoire d'Informatique fondamentale de Lille*
Université de Lille 1, 59655 Villeneuve d'Ascq, France
{urruty, fatima.belkouch, djeraba}@lifl.fr

[**] *University of Massachusetts Boston, Departement of Computer Science*
Boston, Massachusetts 02125, USA
dsim@cs.umb.edu

RÉSUMÉ. *Les systèmes de recherche d'information multimédia deviennent de plus en plus indispensables avec l'émergence des technologies numériques. L'indexation multidimensionnelle est un des problèmes cruciaux dont dépend l'efficacité de ces systèmes. En effet, la description des données multimédia nécessite un très grand nombre de descripteurs, et par conséquent un espace de très grande dimension. L'indexation se heurte au problème connu sous le nom de la "malédiction de la dimension". Nous présentons RPyR (Random Projection classification with Pyramidal Recursive indexing), une technique d'indexation adaptée aux données multidimensionnelles. Celle-ci utilise une technique efficace de classification à base de projections aléatoires et indexe ensuite les données de chaque classe. Nous montrons que les performances de RPyR sont meilleures comparées à d'autres techniques d'indexation multidimensionnelle.*

ABSTRACT. *Multimedia information retrieval systems become increasingly essential with the emergence of numerical technologies. The multidimensional indexing is one of the crucial problems whose effectiveness depends on these systems. Indeed, the description of multimedia data requires a very great number of descriptors, and consequently a high-dimensional data space. In this context, indexing becomes a problem known under the name of the "curse of dimensionality". We present a new indexing technique called RPyR (Random Projection classification with Pyramidal Recursive indexing) and adapted to multidimensional data. This one uses an effective clustering technique based on random projections and an indexing structure for each cluster. We show RPyR outperforms other recent multidimensional indexing techniques.*

MOTS-CLÉS : *Indexation Multidimensionnelle, Classification par projections aléatoires*

KEYWORDS: *Multidimensional Indexing, Random projection clustering*

1. Introduction

Aussi nombreuses que soient les techniques d'indexation proposées dans la littérature, elles demeurent tout de même peu efficaces dans les systèmes de recherche d'information en particulier pour des données multimédia. Les grandes dimensions de l'espace issues du grand nombre de descripteurs rendent problématique l'indexation des données multimédia réelles. Ce problème est connu sous le nom de "la malédiction de la dimension". Les méthodes d'indexation qui ont montré leur limite face à ce problème de dimensionnalité doivent prendre en compte un certain nombre de paramètres à savoir le volume des données, la nature des données, la distribution des données, et assurer un compromis entre la qualité de l'indexation et le temps de réponse pour l'accès aux données.

L'application visée est un moteur de recherche permettant à des professionnels dans la production de films d'entreprise, de rechercher dans de grandes bases de données des séquences vidéo similaires à une description particulière. Exemple, retrouver une séquence filmée dans la nature et contenant une personne. L'indexation passe par : 1) une phase hors ligne qui se doit d'extraire les descripteurs les plus représentatifs des séquences vidéo de la base et d'obtenir des vecteurs numériques préservant le plus fiablement possible la sémantique des séquences. Et ensuite d'indexer ces vecteurs numériques afin d'accélérer la recherche. 2) et une phase de recherche qui consiste d'abord à exprimer une requête, en extraire les descripteurs représentatifs dans un vecteur numérique et ensuite de traduire la similarité par un calcul de proximité entre les descripteurs de la requête et chaque séquence de la base. Nous nous intéressons ici à la phase d'indexation et de recherche. Nous utilisons les requêtes par fenêtrage (Window Query) qui consistent à retourner les vecteurs (séquences) qui sont proches (similaires) à un ϵ près, que nous définissons comme *sélectivité* de la requête. Plus la *sélectivité* est petite plus la recherche est précise. Le choix du paramètre ϵ n'est pas un problème pour notre application destinée à des experts du multimédia, ayant une bonne connaissance de leur base de données et des descripteurs utilisés pour l'indexation.

Dans la littérature, il existe un grand nombre de structures d'indexation arborescentes réparties en deux grandes familles ; l'une basée sur le partitionnement de l'espace de données ([BEN 79]) et l'autre basée sur la répartition des données ([GUT 84]). Plus récemment, d'autres méthodes se sont attaquées au problème de la "malédiction de la dimension" comme la technique de la Pyramide [BER 98], iDistance [YU 01], P+Tree [ZHA 04] qui dépendent généralement des workloads (volume de données, dimension, distribution, etc.) et se sont même vues dépassées par la recherche séquentielle dans certains cas. Dans [URR 05], nous avons proposé une méthode d'indexation qui tire profit des avantages de la technique de pyramide et s'adapte aux différentes distributions de données. Nous avons montré que ses performances sont meilleures comparées aux méthodes citées précédemment. Dans [URR 06a], nous avons approfondi notre étude de performance de ces structures d'indexation et avons analysé les résultats expérimentaux pour montrer que le temps de réponse d'une requête dépend principalement du nombre de données accédées lors de la recherche. Nous avons donc

proposé une nouvelle technique KpyrRec d'indexation multidimensionnelle qui garantit une réduction significative du temps de réponse.

Les structures d'indexation que nous avons proposées utilisent toutes, à la base, k-means[MAC 67] comme algorithme de classification des données. Comme nous n'avons émis aucune hypothèse sur la nature de la distribution des données, k-means semble affecter parfois la performance de la recherche. En effet, le choix du nombre de classes K n'est pas évident, même si l'utilisateur dans notre cas est expérimenté (professionnel des films d'entreprise) et est supposé avoir une connaissance relative quant à la distribution et la nature de ses données. L'algorithme fonctionne mal sur des bases de données contenant du "bruit" dû au fait qu'il tente toujours de construire des classes sphériques. Nous définissons des données "bruit" comme des données réparties aléatoirement dans l'espace sans appartenir à une classe déja existante. Par exemple, supposons que notre base de données de films d'entreprise ne contienne que des vidéos sur l'agriculture et l'industrie en France, alors si on rajoute une vidéo sur la faune en Antarctique, cette vidéo sera considérée comme "bruit". Par conséquent, si les classes sont mal formées, l'espace est par la suite mal partitionné, et la fenêtre WQ risque d'approcher inutilement des classes, ce qui va élargir l'espace de recherche et par conséquent allonger le temps de réponse.

Pour ces raisons, nous avons proposé une nouvelle méthode de partitionnement de l'espace mieux adaptée aux grands espaces. Elle combine une nouvelle méthode de classification utilisant le principe des projections aléatoires et les méthodes de classification basée sur la densité. Chaque classe résultat est ensuite indexée, y compris la classe des données bruit. L'algorithme proposé projette les points dans un espace unidimensionnel choisi aléatoirement et ensuite agrège les projections pour retrouver les classes. Avec une complexité de l'ordre de $O(N \log N)$, où N est le nombre de points de l'espace, notre technique de classification apporte une bonne classification de nos données. Nous justifions avec nos expérimentations le fait qu'une meilleure classification avantage considérablement la recherche.

Ce papier est organisé comme suit : La section 2 présente quelques travaux existant sur l'indexation ainsi que notre méthode KpyrRec. La section 3 résume ses performances et en tire des conclusions d'amélioration. La section 4 présente dans un premier temps notre technique de classification par projection aléatoire et dans un second temps, détaille notre nouvelle méthode d'indexation multidimensionnelle *RPyR*. La section 5 discute les performances de la recherche. La section 6 conclut l'article.

2. Indexation multidimensionnelle

2.1. *Quelques travaux dans le domaine*

L'étude des structures d'indexation multidimensionnelle a émergé avec l'apparition des structures utilisant des arbres hiérarchiques comme le k-d tree [BEN 79] et les nombreuses déclinaisons du R-tree [GUT 84]. Ces structures et leurs variantes ont montré leur efficacité mais sont dépassées pour les grandes dimensions. Ce phéno-

mène est connu sous le nom de la "malédiction de la dimension". Deux méthodes apparaissent en 1998 pour faire face à cette malédiction, le VA-File [WEB 98] et la technique de la pyramide [BER 98]. Le Va-File est une méthode basée sur une approximation de l'espace de données par codage. Inspirées par cette méthode, d'autres méthodes sont apparues comme le IQ-Tree [BER 00] et le LPC-File [CHA 02]. Ces méthodes ont de bonnes performances pour une répartition de données homogènes mais sont dépassées par des données hétérogènes de grandes dimensions. La technique de la pyramide est la première des méthodes basées sur un découpage de l'espace de données pour représenter les points multidimensionnels par des index ou valeur clé à une dimension. Plusieurs méthodes se sont inspirées de la technique de la pyramide, telles que iMinMax [OOI 00], Idistance [YU 01] et P+Tree [ZHA 04]. Nous nous sommes plus intéressés à cette catégorie de méthodes et nous avons proposé deux méthodes Kpyr [URR 05] et son amélioration KpyrRec [URR 06a].

Une synthèse de l'ensemble des méthodes de la catégorie "découpage géométrique de l'espace de données"' a été présentée en deux tableaux dans [URR 06a]. Le premier tableau donne les caractéristiques de chaque méthode d'indexation selon différents critères : la dimension, le nombre et la distribution des données traitées, ainsi que leur adaptation aux deux types de requêtes KNN (requête par plus proches voisins) et WQ (requête par fenêtrage). Le deuxième tableau synthétise les avantages et inconvénients de chacune de ces méthodes.

2.2. *Notre structure d'indexation KpyrRec*

Une étude approfondie de la technique de la pyramide nous a conduit à réfléchir sur la façon de réunir les conditions favorables à son application. Nous avons ainsi proposé une meilleure technique d'indexation PyrRec qui réorganise efficacement l'espace en sous espaces homogènes et introduit une nouvelle structure qui gère ces espaces.

L'idée générale de PyrRec est de faire un découpage de l'espace équilibré en volume de données. La méthode applique en premier un découpage de l'espace en pyramides. Chaque pyramide est découpée ensuite en *nt* tranches contenant le même nombre de données. Nous déterminons ces tranches en calculant les hauteurs respectives h_i et h_{i+1} de la pyramide.

$$h_i = \frac{h * (\sqrt[D]{i} - \sqrt[D]{i-1})}{\sqrt[D]{Nt}}, \qquad [1]$$

avec $1 \leq i < Nt$, D : nombre de dimensions, h : hauteur des pyramides

Nous proposons de faire une récurrence de la technique de pyramide sur l'ensemble des données appartenant à la tranche. Chaque récurrence réduit le nombre de dimension de 1. Nous nous fixons un seuil, correspondant au nombre de données maximum par tranche, pour arrêter la récurrence. En général, le nombre de récurrence ne dépasse pas 3 même avec un petit seuil. Par exemple, pour 1 000 000 données de dimension 100, avec un seuil de 10 données, nous avons en moyenne 5 000 données

par pyramide pour la première récurrence, ce qui nous donne environ 25 données par pyramide pour une deuxième récurrence. Si on découpe chaque pyramide par exemple en 100 tranches, une troisième récurrence n'est nécessaire que dans des cas de distribution de données très hétérogènes avec des régions de l'espace très denses.

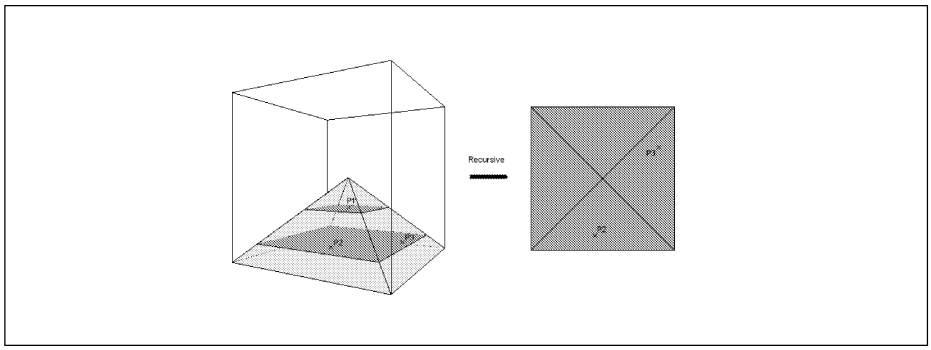

Figure 1. *Représentation géométrique de PyrRec*

La figure 1 nous montre un exemple d'utilisation d'une récurrence. Supposons que la tranche de P1 ne subisse pas de récurrence, le point P1 aura un seul index. P2 et P3 sont dans une tranche qui subit une récurrence, ils auront un index principal similaire et un deuxième index créé à partir du découpage sur un espace de dimension D-1 (partie droite de la figure 1). Ce deuxième index distingue les deux points P2 et P3. La recherche pourra donc être affinée par cette récursivité.

L'algorithme PyrRec : Pour construire les index des points de l'espace de données, PyrRec procède comme suit :

– Pour chaque point, il détermine la pyramide *cp* qui contient le point, en se basant sur des calculs de distances.

– Ensuite, en fonction de la hauteur du point par rapport au sommet de la pyramide, il détermine la tranche *tp* de la pyramide contenant le point. A partir de *cp* et *tp*, il crée l'index principal (un entier) du point.

– L'algorithme est récursif : si le nombre de points dans la tranche dépasse un certain seuil on ré-applique l'algorithme sur les données de la tranche, un espace de données dont la dimension est réduite de 1.

Nous obtenons alors un ou plusieurs index pour chaque point que l'on gardera en mémoire avec l'index principal.

L'algorithme final KpyrRec est l'association de k-means avec PyrRec. On divise l'espace de données en K classes homogènes avec l'algorithme de k-means puis on applique PyrRec sur chaque classe pour obtenir K arbres B+. Ces derniers sont répartis dans un arbre *Espace* binaire en fonction de la position des classes dans l'espace de données. L'algorithme de recherche est similaire à celui de RPyR, présenté dans la section 4.

3. Performances de KPyrRec

L'implémentation de KPyrRec a été réalisée en langage Java. L'évaluation des performances est faite sur un ordinateur AMD 3Ghz avec 1Go de RAM. Toutes les expérimentations ont été faites sur des données synthétiques. Dans [URR 06a] nous avons analysé en détail nos résultats mesurant le temps de réponse total pour une requête WQ sous différentes caractéristiques de la base de données : la taille de la base de données, la dimension des données et la sélectivité (largeur de la fenêtre de WQ : précision de la requête).

Nous avons d'abord comparé la technique de la pyramide à notre méthode PyrRec et avons évalué le gain en temps pour des recherches sur un ensemble de données homogènes. La figure 2 présente le temps de réponse total en fonction de la sélectivité pour 1 000 000 données de dimension 80. Nous avons utilisé pour ces courbes de résultats trois récurrences au maximum. La réduction du temps de réponse est assez significative, allant jusqu'à 50%. D'où, l'intérêt de remplacer la technique de la pyramide avec PyrRec dans l'algorithme KPyrRec.

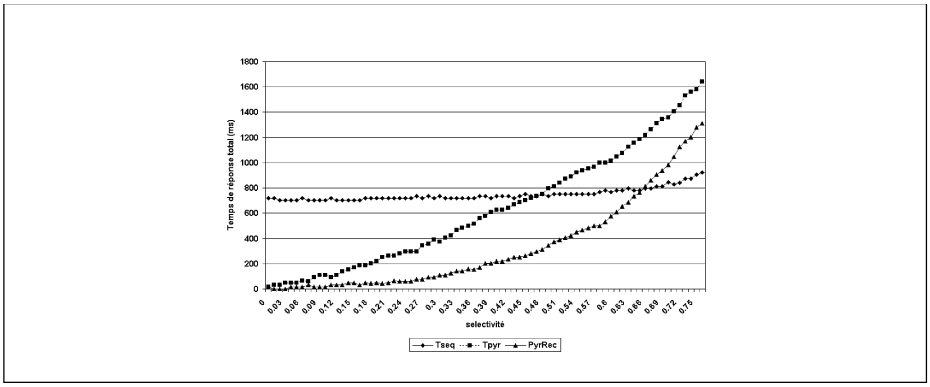

Figure 2. *Effet de la sélectivité avec 1 000 000 données*

Nous nous sommes ensuite intéressés aux performances de la méthode globale KpyrRec sur des données hétérogènes. Nous avons montré que KpyrRec obtient de bonnes performances en temps de réponse. La figure 3 nous montre que pour 50 dimensions et 1 000 000 données réparties en 4 classes, KpyrRec obtient un meilleur temps de réponse quelque soit la sélectivité de la requête par fenêtrage comparée aux autres structures d'indexation : recherche séquentielle, Pyramide, P+Tree et Kpyr.

En résumé : Le résultat important que nous avions constaté au cours de nos expérimentations précédentes était le lien existant entre le temps de réponse et le nombre de données accédées pour une recherche. Nous avons donc introduit PyrRec, une approche récurrente et approximative de la technique de pyramide. Comme nous l'avons montré dans [URR 06a], l'amélioration du temps de réponse a été constaté pour différents paramètres de la base de données : nombre de classes, taille de données, dimension, etc. En effet, l'utilisation de PyrRec réduit le temps de réponse à une requête car

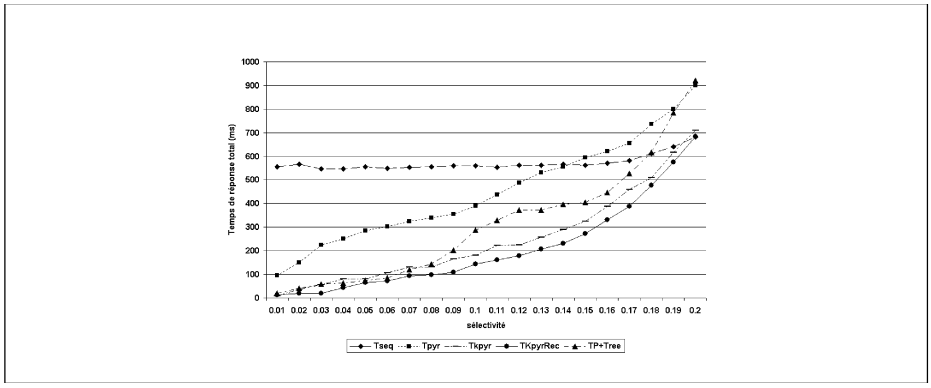

Figure 3. *Effet de la sélectivité avec 1 000 000 données*

une recherche accède à un nombre de données très réduit par rapport à une recherche utilisant la technique de la Pyramide. Nous avons aussi montré que le coût mémoire supplémentaire des index récurrents produits par PyrRec était négligeable.

Problèmes : Les structures d'indexation que nous avons proposées, sont basées sur l'algorithme k-means [MAC 67] en $O(N^2)$pour la classification des données. Comme nous n'avons émis aucune hypothèse sur la nature de la distribution des données, k-means semblait affecter parfois la performance de la recherche. L'algorithme k-means ne gère que des classes de formes sphériques. Si la répartition des données devient plus complexe, ou si la base de données contient des données "bruit", k-means donnera des résultats très aléatoires, voire incorrects. De plus, k-means présente le problème du choix du nombre de classes K qui demeure tout de même problématique, même pour un utilisateur expérimenté de la base de données. Pour une structure d'indexation multidimensionnelle, la classification des données n'affecte pas la qualité des résultats de la recherche, mais elle peut en revanche influencer le temps de réponse. En effet, si les classes sont mal formées, la requête risque d'atteindre inutilement des classes, ce qui va augmenter le nombre de données accédées inutilement et par conséquence le temps de réponse.

Pour ces raisons, nous proposons RPyR, une nouvelle méthode d'indexation multidimensionnelle alliant une méthode de classification des données, plus adaptée à différentes répartitions des données et gérant dans le même temps la possibilité d'avoir des données "bruit", à la technique PyrRec.

4. RPyR : notre nouvelle méthode d'indexation multidimensionnelle

Dans une première partie, nous donnons un aperçu des méthodes de classification puis nous présentons KPyR, notre nouvelle méthode d'indexation multidimensionnelle. KPyR tente de remédier aux problèmes soulevés par différentes répartitions des données ainsi qu'aux données "bruit".

4.1. *Aperçu de méthodes de classification*

La classification est un sujet très actif dans le domaine de la fouille de données. Les algorithmes de classification y sont présents pour de nombreuses applications ([ZAÏ 02, DJE 03]), incluant la fouille de données multimédia. L'état de l'art de ce domaine est exposé dans plusieurs articles de synthèse apparaissant avec une certaine régularité au fil des ans (voir [JAI 88, JAI 99, TAN 06]).

La classification dans des espaces de données de petites dimensions est relativement simple. Par exemple dans un espace uni-dimensionnel, les régions de densité élevée sont facilement identifiables par une simple recherche linéaire. Avec l'augmentation de la dimensionnalité, le problème devient plus complexe. La notion de classification par projection est alors introduite par Agrawal et al. dans [AGR 98]. Ces derniers ont observé que les points dans des sous espaces de plus faible dimensionnalité peuvent mieux se classifier que dans l'ensemble de l'espace $\mathbb{R}^n$, idée qu'ils ont utilisée dans leur algorithme CLIQUE.

Dans [AGG 99] Aggarwal et al. ont développé l'algorithme PROCLUS basé sur la recherche des classes dans des sous espaces de petites dimensions. Leur idée principale vient de l'utilisation du lemme de Johnson-Lindenstrauss [JOH 84]. Ce lemme affirme qu'un ensemble de données dans un espace multidimensionnel euclidien peut être projeté dans un sous espace de dimension moins élevée tel que la distance entre deux des points de cet ensemble varie d'un facteur multiplicatif $1+\epsilon$, pour $\epsilon \in [0, 1]$. Des preuves de ce résultat sont proposées par Frankl et Maehara [FRA 88] et par Dasgupta et Gupta [DAS 99]. Par ailleurs, des travaux de Indyk et al.([GIO 99] et [IND 00]) utilisent des techniques de réduction de dimension, telles que LSH pour répondre au problème des plus proches voisins.

Un autre algorithme de classification par projection est proposé par [AGA 04], une extension de l'algorithme k-means dans des sous espaces arbitraires où une fonction objective est introduite qui a pour rôle de créer un compromis entre la dimension d'un sous espace et son erreur de classification.

4.2. *Notre méthode de classification par projection aléatoire*

Nous proposons ici un algorithme de classification qui combine des idées provenant des projections aléatoires et de la classification par densité. Notre méthode se rapproche de celle proposée par [AGR 98]. Nous construisons des classes dans des espaces de plus petites dimensions et nous sélectionnons les dimensions qui peuvent identifier au mieux les classes dans notre base de données originale. Notre contribution principale consiste à choisir un repère orthonormal aléatoire dans $\mathbb{R}^n$. Nous projetons ensuite l'ensemble des points de notre base de données sur chaque axe de ce repère. Nous avons montré dans [URR 06b] que choisir un repère aléatoire à la place du repère original diminue les risques d'avoir un phénomène d'*occultation* entre les classes. Le phénomène d'*occultation* est le cas où les projections de deux classes dis-

tinctes dans l'espace d'origine ont une intersection commune dans le sous espace de projection. Un exemple d'*occultation* entre deux classes sur une projection aléatoire est représenté dans la figure 4.

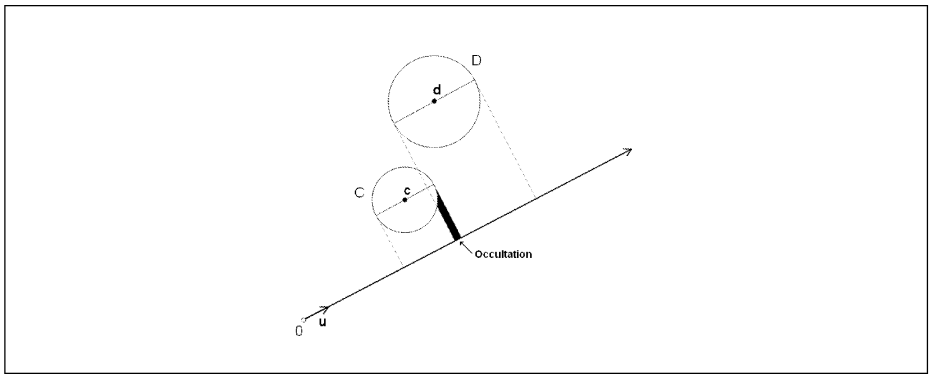

Figure 4. *Phénomène d'occultation entre deux classes sur une projection aléatoire*

Les histogrammes de densité des projections unidimensionnelles contenant le plus d'informations sur les classes sont ensuite combinés pour retrouver la localisation des classes dans l'espace de données original. Une deuxième partie de notre algorithme affine la classification trouvée par projection aléatoire, à l'aide de deux processus distincts : la *bi-modulation* et *l'expansion des classes*.

A partir d'une base de données de dimension n, la phase de projections aléatoires suit 5 étapes :

– choix d'une nouvelle base orthonormale aléatoire $n * n$, en utilisant une distribution uniforme sur un intervalle puis en appliquant ensuite la technique de Gram-Schmidt ;

– projection de chaque point sur chaque dimension de la nouvelle base choisie, ce qui permet d'obtenir n histogrammes de densité, voir figure 5 ;

– choix des 10% "meilleurs" histogrammes contenant le plus d'informations sur les classes. La qualité d'une projection est évaluée par le produit de la moyenne des hauteurs des pics et du nombre de pics d'un histogramme. Dans l'exemple de la figure 5, les projections 14 et 15 ont visiblement 3 hauts pics, et sont ainsi jugées de meilleure qualité que les autres ;

– extraction d'une liste d'intervalles dans lesquels la densité du nombre de points est élevée, à partir des "meilleurs" histogrammes sélectionnés. Ces intervalles pourraient contenir une ou plusieurs classes. Dans l'exemple figure 6, les trois intervalles obtenus pour la projection 14 sont compris entre 9 et 11, 21 et 28 et enfin 35 et 43.

– combinaison de l'ensemble de ces intervalles pour former des régions de densité élevée que nous définissons ensuite comme les classes obtenues par notre phase de projection aléatoire. La figure 7 montre la combinaison de la projection 14 et de la projection 15. Cette combinaison nous permet de connaître 9 régions (en gris foncé

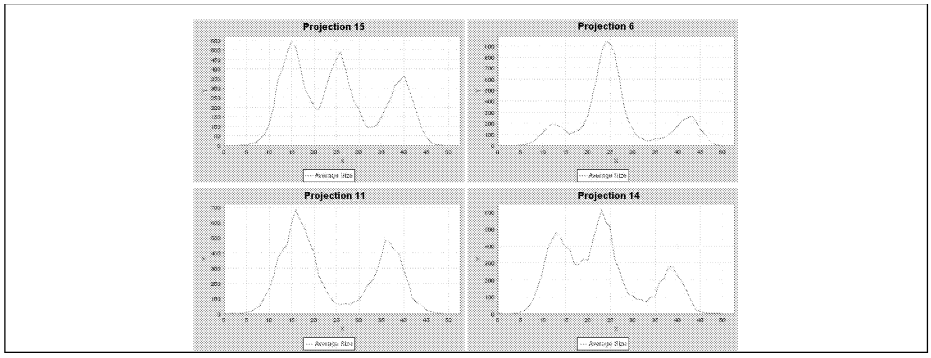

Figure 5. *Exemples d'histogrammes de densité sur certaines projections*

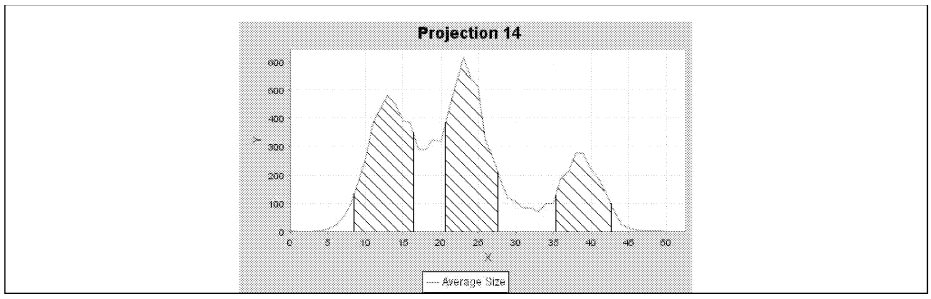

Figure 6. *Exemple d'intervalles choisis pour la projection 14*

sur la figure) qui pourraient contenir une densité élevée de données, il suffit ensuite de vérifier pour chaque région le nombre de point qu'elle contient.

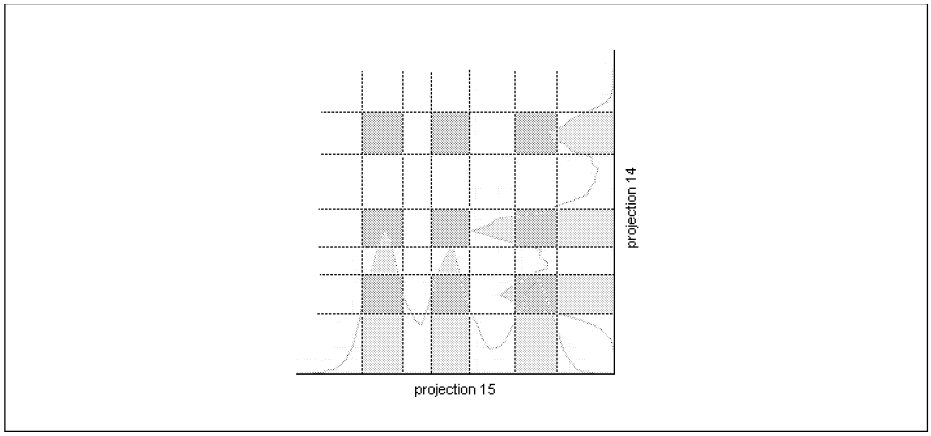

Figure 7. *Exemple de combinaison d'intervalles pour deux projections*

Dans [URR 06b], nous fournissons plus de détails sur le choix des "meilleurs" histogrammes et sur l'algorithme de combinaisons des histogrammes pour former notre classification. Nous démontrons aussi que le coût asymptotique de notre algorithme de classification par projection aléatoire est en $O(N \log N)$, où N est le nombre de données de notre base. Cependant, il est important de noter que si le nombre de dimension n est relativement élevé par rapport au nombre de données, nous aurons alors un algorithme en $O(n^2N + N \log N)$.

Une deuxième phase de notre algorithme de classification consiste en deux post-processus qui permettent d'améliorer la classification obtenue par projection aléatoire. Cette deuxième partie n'affecte pas le coût asymptotique de notre algorithme.

– Le premier post-processus appelé *bi-modulation* consiste en une fusion de deux classifications, chacune obtenue par l'exécution de la première phase de projections aléatoires. Chaque exécution fournit des classes et un ensemble assez important de points non classés. L'idée principale est de combiner ces deux classifications pour en obtenir une meilleure et diminuer l'ensemble des points non classés.

– Le deuxième post-processus est une expansion d'un paramètre epsilon des régions des classes obtenues. Cette expansion permet d'inclure les points non classés se trouvant à proximité d'une classe dans cette dernière. Les points restants non classés après ces deux post-processus sont considérés comme "bruit" par notre algorithme.

4.3. *Algorithme d'indexation et de recherche de RPyR*

Notre méthode de classification propose une solution pour répondre aux problèmes de la méthode k-means : le choix du nombre de classes K, la possibilité de gérer des bases de données bruitées et trouver des classes pas seulement de forme sphérique.

Nous proposons une nouvelle technique d'indexation RPyR qui tire profit des avantages de notre méthode de classification par projection aléatoire et de PyrRec proposée l'année dernière dans [URR 06a] et présentée brièvement dans la section 2.

L'**algorithme d'indexation** se divise en plusieurs étapes :

– K classes et un ensemble de points non classés que nous considérons comme du bruit sont obtenus avec notre méthode de classification par projection aléatoire.

– PyrRec indexe chaque classe trouvée dans un arbre B+.

– Les données "bruit" sont indexées comme une classe dans un arbre B+.

– Les $K + 1$ arbre B+ sont stockés en mémoire avec leur rectangle minimum englobant et leur "médoïde", le point le plus proche du centre de gravité de la classe, pour un accès rapide aux bonnes classes lors de la recherche.

L'**algorithme de recherche** procède comme suit :

– A partir d'un point requête q, une "fenêtre" requête est formée en fonction de la sélectivité choisie.

Algorithm	VP	FP	VN	FN	Précision	Rappel	F1
k-means	62	28	10	1	0.69	0.98	0.81
Notre algorithme	63.2	0.1	28.9	7.8	0.99	0.89	0.94

Tableau 1. *Tableau de contingence entre notre algorithme et k-means*

– Tous les arbres B+ concernés par la recherche sont trouvés grâce à la position de la requête et les informations conservées par chaque arbre B+ (représentant une classe).

– Pour chaque classe atteinte, la requête est transformée et détermine les tranches des pyramides qu'il faut scanner.

– Enfin, avec un accès aux feuilles de l'arbre B+, les données "résultats" sont récupérées. Dans le cas où un point aurait des index multiples, un simple test de comparaison par index suffit pour savoir si le point est à l'intérieur ou à l'extérieur de la fenêtre requête.

Nous remarquons que les opérations supplémentaires et nécessaires lors d'une recherche dans notre arbre sont deux tests sur des nombres entiers pour les points ayant plusieurs index. Ce coût est négligeable par rapport au coût du traitement inutile des données gagné grâce à la récurrence.

5. Performances

5.1. *Notre méthode de classification*

Nous présentons rapidement quelques résultats obtenus sous forme d'un tableau de contingence (voir Tableau 1) pour notre algorithme de classification comparé à l'algorithme k-means, seule comparaison intéressante pour notre technique d'indexation, sur une base de données à répartition hétérogène de 10000 points de 80 dimensions et sur la même base de données contenant 10% de bruit. Le tableau de contingence est un tableau de mesures statistiques calculées à partir des classes obtenues par un algorithme de classification et des classes réelles. On note VP pour les Vrais Positifs, FP pour les Faux Positifs, VN pour les Vrais Négatifs et FN pour les Faux Négatifs.

De nombreux autres tests sont présentés dans [URR 06b] justifiant les choix expérimentaux des paramètres de l'algorithme. Les résultats obtenus montrent aussi que le nombre de dimension et le volume de données influent peu sur les résultats. Notons aussi que pour une base de données non bruitée repartie en classes sphérique, notre algorithme n'offre aucun avantage par rapport à k-means. Une application particulière sur des images réelles présente des résultats visuellement corrects. L'ensemble des expérimentations montre les bonnes performances de notre méthode de classification par projections aléatoires par rapport à k-means. Rappelons que notre algorithme en $O(N \log N)$ est moins complexe que celui de k-means.

5.2. *Performances de RPyR*

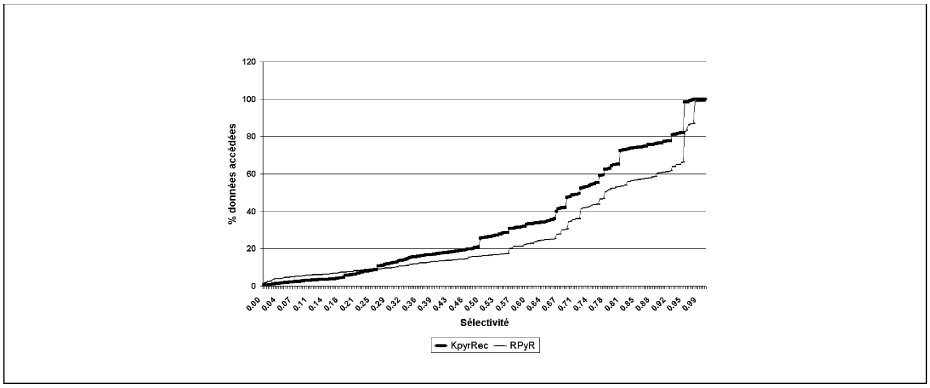

Figure 8. *% données accédées en fonction de la sélectivité pour une base de données de dimension 20*

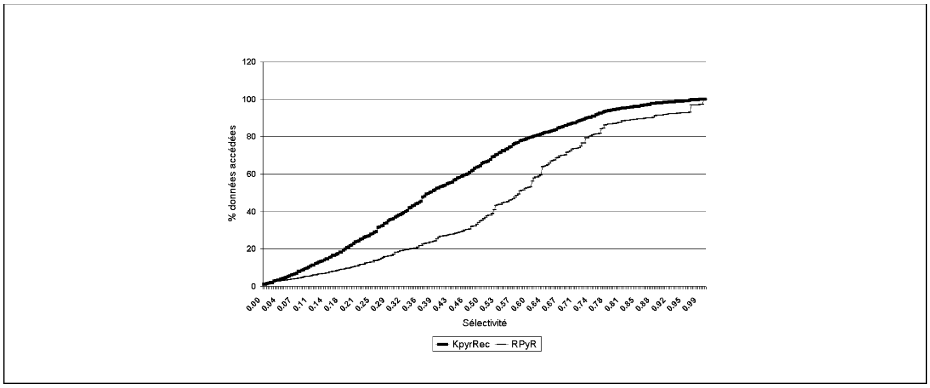

Figure 9. *% données accédées en fonction de la sélectivité pour une base de données de dimension 20 avec 5% de bruit*

Par ailleurs, nous avons présenté en 2006 de nombreux résultats comparant notre méthode d'indexation multidimensionnelle KpyrRec à quelques méthodes de l'état de l'art ainsi qu'à la recherche séquentielle. Certains de ces résultats sont présentés section 3. L'une des conclusion les plus importantes que nous avons pu faire est que le temps de réponse total d'une requête par fenêtrage est proportionnel au nombre de données accédées. C'est pourquoi nous présentons dans cette section des courbes du pourcentage de données accédées par rapport au nombre total de points dans notre base de données.

Les figures 8 à 11 montrent une étude comparative entre le % de données accédées par KpyrRec et notre nouvelle méthode RPyR sur une base de données synthétique contenant 100000 points répartis en 20 classes de formes aléatoires. Les figures 8

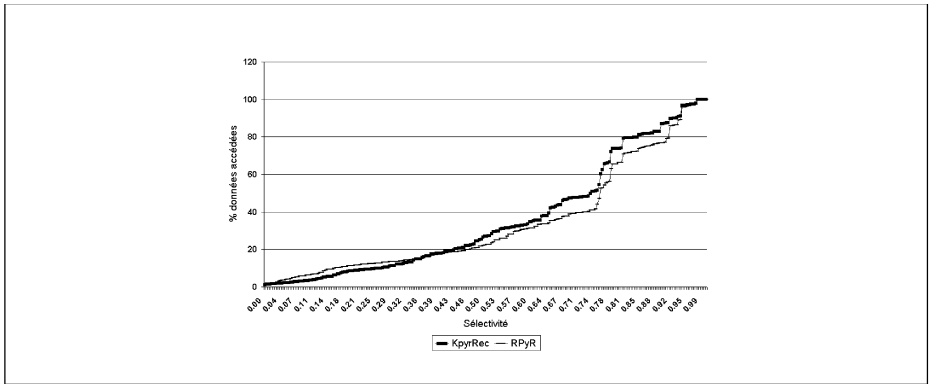

Figure 10. *% données accédées en fonction de la sélectivité pour une base de données de dimension 80*

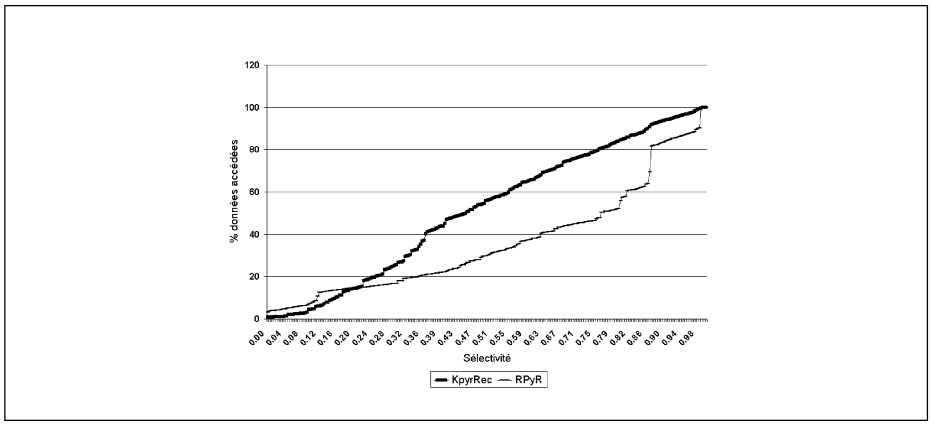

Figure 11. *% données accédées en fonction de la sélectivité pour une base de données de dimension 80 avec 5% de bruit*

et 9 montrent les expériences effectuées sur une base de données de 20 dimensions alors que les figures 10 et 11 montrent celles effectuées sur une base de données de 80 dimensions. Pour les figures 9 et 11, nous avons rajouté 5% de données "bruit" réparties aléatoirement dans l'ensemble de l'espace de données.

L'ensemble de ces expérimentations montre l'avantage considérable apporté par l'utilisation de notre algorithme de projection aléatoire par rapport à k-means, nous obtenons une baisse du volume de données accédées comprise en moyenne entre 10% et 50% entre KpyrRec et RPyR. L'algorithme de recherche étant le même pour les deux méthodes, le temps de recherche est proportionnel à ces résultats. Une meilleure classification des données et une indexation adaptée améliore la rapidité d'une requête par une diminution d'accès à des données inutiles.

6. Conclusion

Nous avons présenté, dans cet article, RPyR, une nouvelle méthode d'indexation multidimensionnelle combinant une méthode de classification par projection aléatoire à la technique d'indexation pyramidale récursive PyrRec. Cette association rend notre méthode plus robuste à des répartitions des données classables mais de forme pas nécessairement sphérique et gérant dans le même temps la possibilité d'avoir des données "bruit". Nos expérimentations ont mis en valeur les performances de RPyR par rapport à KpyrRec. En effet le nombre de données accédées lors une requête a nettement diminué ce qui prouve une bonne classification en $O(N \log N)$ par notre méthode par rapport à celle de KpyrRec en $O(N^2)$. Dans un futur proche, nous continuerons nos recherches et expérimentons sur RPyR en l'appliquant sur différentes bases de données réelles afin de tester notre algorithme sur des répartitions de données aléatoires.

7. Bibliographie

[AGA 04] AGARWAL P., MUSTAFA N. H., « k-Means Projective Clustering », *Proceedings of PODS*, 2004, p. 155–165.

[AGG 99] AGGARWAL C. C., PROCOPIUC C., WOLF J. L., YU P. S., PARK J. S., « Fast algorithms for projected clustering », *Proceedings of ACM-SIGMOD Conference on Management of Data*, 1999, p. 61–72.

[AGR 98] AGRAWAL R., GEHRKE J., GUNOPULOS D., RAGHAVAN P., « Automatic subspace clustering of high dimensional data for data mining applications », *Proceedings of the ACM-SIGMOD Int. Conf. Management of Data,*, 1998, p. 94–105.

[BEN 79] BENTLEY J. L., « Multidimensional Binary Search in Database Application », , 1979, p. 333–340.

[BER 98] BERCHTOLD S., BOHM C., KRIEGEL H. P., « The Pyramid Technique : Towards Breaking the Curse of Dimensionality », *Proceedings of ACM SIGMOD Int. Conf on Management of Data*, Seatle,Washington,USA, 1998, p. 142–153.

[BER 00] BERCHTOLD S., BÖHM C., JAGADISH H. V., KRIEGEL H. P., SANDER J., « Independant Quantization : An Index Compression Technique for High-dimensional Data Spaces », *Proceedings of 16-th Int. Conf. on Data Engineering, IEEE ICDE*, San diego, California, USA, 2000, p. 577–588.

[CHA 02] CHA G. H., X. ZHU D. P., CHANG C. W., « An Efficient Indexing Method for Nearest Neighbors Searches in High Dimensional Image Databases », vol. 4, 2002, p. 76–87.

[DAS 99] DASGUPTA S., GUPTA A., « An Elementary Proof of the Johnson-Lindenstrauss Lemma », rapport n° TR-99-006, 1999, International Computer Science Institute.

[DJE 03] DJERABA C., Ed., *Multimedia Mining - A Highway to Intelligent Multimedia Documents*, Kluwer, Boston, 2003.

[FRA 88] FRANKL P., MAEHARA H., « The Johnson-Lindenstrauss Lemma and the Sphericity of Some Graphs », *J. Comb. Theory B*, vol. 44, 1988, p. 355–362.

[GIO 99] GIONIS A., INDYK P., MOTWANI R., « Similarity Search in High Dimensions via Hashing. », *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland, UK, 1999, p. 518-529.

[GUT 84] GUTTMAN A., « R-Trees : a Dynamic Index Structure for Spatial Searching », *Proceedings of ACM SIGMOD*, Boston, USA, 1984, ACM Press, p. 47–57.

[IND 00] INDYK P., « Dimensionality reduction techniques for proximity problems », *Symposium on Discrete Algorithms*, 2000, p. 371-378.

[JAI 88] JAIN A. K., DUBES R., *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[JAI 99] JAIN A. K., MURTY M. N., FLYNN P. J., « Data Clustering : A Review », *ACM Computing Surveys*, vol. 31, 1999, p. 264–323.

[JOH 84] JOHNSON W. B., LINDENSTRAUSS J., « Extensions of Lipshitz mappings into Hilbert Spaces », *Contemporary Mathematics*, vol. 26, 1984, p. 189–206.

[MAC 67] MACQUEEN J. B., « Some Methods for classification and Analysis of Multivariate Observations », *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, 1967, University of California Press, p. 281–297.

[OOI 00] OOI B. C., TAN K. L., YU C., BRESSAN S., « Indexing the Edges - a Simple and Yet Efficient Approach to High-dimensional », *Proceedings of 19-th ACM SIGMOD SIGACT SIGART Symposium on Principles of Database Systems*, Dallas, Texas, USA, 2000, p. 166–174.

[TAN 06] TAN P. N., STEINBACH M., KUMAR V., *Introduction to Data Mining*, Pearson/Addison-Wesley, Boston, 2006.

[URR 05] URRUTY T., BELKOUCH F., DJERABA C., « Kpyr, une Structure Efficace d'Indexation de Documents Vidéo », *21ème journées Informatique des Organisation et Systèmes d'Information et de Décision (Inforsid05)*, Grenoble, France, 2005, p. 403–418.

[URR 06a] URRUTY T., BELKOUCH F., DJERABA C., « Indexation Multidimensionnelle : KpyrRec, une Amélioration de Kpyr », *22ème journées Informatique des Organisation et Systèmes d'Information et de Décision (Inforsid06)*, Hammamet, Tunisie, 2006, p. 831–846.

[URR 06b] URRUTY T., DJERABA C., SIMOVICI D., « Clustering by Random Projections : Application to Image Segmentation », *Proceedings of 7-th Int. Workshop on Multimedia Data Mining "Merging Multimedia and Data Mining Research" (ACM KDD/MDM 2006*, Philadelphia, USA, 2006, p. 119–124.

[WEB 98] WEBER R., SCHEK H. J., BLOTT S., « A Quantitative Analysis and Performance Study for Similarity Search Methods in High-dimensional Spaces », *Proceedings of 24-th Int. Conf. on Very Large Data Bases*, New York, USA, 1998, p. 194–205.

[YU 01] YU C., OOI B. C., TAN K. L., JAGADISH H. V., « Indexing the Distance : an Efficient Method to KNN Processing », *Proceedings of 27-th Int. Conf. on Very Large Data Bases*, Rome,Italy, 2001, p. 421–430.

[ZAÏ 02] ZAÏANE O. R., SIMOFF S., DJERABA C., Eds., *Mining Multimedia and Complex Data*, vol. LNAI 2797 de *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 2002.

[ZHA 04] ZHANG R., OOI B. C., TAN K. L., « Making the Pyramid Technique Robust to Query Types and Workloads », *Proceedings of 20-th Int. Conf. on Data Engineering, IEEE ICDE*, Boston, USA, 2004, p. 313–324.